**Dining Philosopher’s Problem :**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (phnum + 4) % N

#define RIGHT (phnum + 1) % N

int state[N];

int phil[N] = { 0, 1, 2, 3, 4 };

sem\_t mutex;

sem\_t S[N];

void test(int phnum)

{

if (state[phnum] == HUNGRY

&& state[LEFT] != EATING

&& state[RIGHT] != EATING) {

// state that eating

state[phnum] = EATING;

sleep(2);

printf("Philosopher %d takes fork %d and %d\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is Eating\n", phnum + 1);

// sem\_post(&S[phnum]) has no effect

// during takefork

// used to wake up hungry philosophers

// during putfork

sem\_post(&S[phnum]);

}

}

// take up chopsticks

void take\_fork(int phnum)

{

sem\_wait(&mutex);

// state that hungry

state[phnum] = HUNGRY;

printf("Philosopher %d is Hungry\n", phnum + 1);

// eat if neighbours are not eating

test(phnum);

sem\_post(&mutex);

// if unable to eat wait to be signalled

sem\_wait(&S[phnum]);

sleep(1);

}

// put down chopsticks

void put\_fork(int phnum)

{

sem\_wait(&mutex);

// state that thinking

state[phnum] = THINKING;

printf("Philosopher %d putting fork %d and %d down\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is thinking\n", phnum + 1);

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void\* philospher(void\* num)

{

while (1) {

int\* i = num;

sleep(1);

take\_fork(\*i);

sleep(0);

put\_fork(\*i);

}

}

int main()

{

int i;

pthread\_t thread\_id[N];

// initialize the semaphores

sem\_init(&mutex, 0, 1);

for (i = 0; i < N; i++)

sem\_init(&S[i], 0, 0);

for (i = 0; i < N; i++) {

// create philosopher processes

pthread\_create(&thread\_id[i], NULL, philospher, &phil[i]);

printf("Philosopher %d is thinking\n", i + 1);

}

for (i = 0; i < N; i++)

pthread\_join(thread\_id[i], NULL);

}

**Producer**

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<semaphore.h>

#include<unistd.h>

int c=0;

sem\_t full,mutex,empty;

void \*producer(){

sem\_wait(&empty);

sem\_wait(&mutex);

printf("producer has produced item %d\n",++c);

sem\_post(&mutex);

sem\_post(&full);

}

void \*consumer(){

sem\_wait(&full);

sem\_wait(&mutex);

printf("consumer has consumed item %d\n",c--);

sem\_post(&mutex);

sem\_post(&empty);

}

int main(){

pthread\_t pid[5] , cid[5];

sem\_init(&empty,0,5);

sem\_init(&full,0,0);

sem\_init(&mutex,0,1);

int i,m,n;

printf("enter numberof producer");

scanf("%d",&m);

printf("enter numberof consumer");

scanf("%d",&n);

for(i=0;i<m;i++)

pthread\_create(&pid[i],NULL,producer,NULL);

for(i=0;i<n;i++)

pthread\_create(&cid[i],NULL,consumer,NULL);

for(i=0;i<m;i++)

pthread\_join(pid[i],NULL);

for(i=0;i<n;i++)

pthread\_join(cid[i],NULL);

exit(0);

}